

A Theoretical Assessment of Solution Quality in Evolutionary Algorithms for the Knapsack Problem

Jun He, Boris Mitavskiy and Yuren Zhou

Abstract

Evolutionary algorithms are well suited for solving the knapsack problem. Some empirical studies claim that evolutionary algorithms can produce good solutions to the 0-1 knapsack problem. Nonetheless, few rigorous investigations address the quality of solutions that evolutionary algorithms may produce for the knapsack problem. The current paper focuses on a theoretical investigation of three types of $(N+1)$ evolutionary algorithms that exploit bitwise mutation, truncation selection, plus different repair methods for the 0-1 knapsack problem. It assesses the solution quality in terms of the approximation ratio. Our work indicates that the solution produced by pure strategy and mixed strategy evolutionary algorithms is arbitrarily bad. Nevertheless, the evolutionary algorithm using helper objectives may produce $1/2$ -approximation solutions to the 0-1 knapsack problem.

Index Terms

Evolutionary algorithm, approximation algorithm, knapsack problem, solution quality

I. INTRODUCTION

The knapsack problem is an NP-hard combinatorial optimisation problem [1], which includes a variety of knapsack-type problems such as the 0-1 knapsack problem and multi-dimensional knapsack problem. In the last two decades, evolutionary algorithms (EAs), especially genetic algorithms (GAs), have been well-adopted for tackling the knapsack problem [2]–[5]. The problem has received a particular interest from the evolutionary computation community for the following two reasons. The first reason is that the binary vector representation of the candidate solutions is a natural encoding of the 0-1 knapsack problem's search space. Thereby, it provides an ideal setting for the applications of genetic algorithms [6]. On the other hand, the multi-dimensional knapsack problem is a natural multi-objective optimization problem, so that it is often taken as a test problem for studying multi-objective optimization evolutionary algorithms (MOEAs) [7]–[11].

A number of empirical results in the literature (see, for instance, [7]–[12]) assert that EAs can produce “good” solutions to the knapsack problem. A naturally arising question is then how to measure the “goodness” of solutions that EAs may produce? To address the question, the most popular approach is to compare the quality of the solutions generated by EAs via computer experiments. For example, the solution quality of an EA is measured by the best solution found within 500 generations [6]. Such a comparison may help to compare performance of different EAs, yet it seldom provides any information regarding the proximity of the solutions produced by the EAs to the optimum.

From the viewpoint of algorithm analysis, it is important to assess how “good” a solution is in terms of the notion of approximation ratio (see [13]). There are several effective approximation algorithms for solving the knapsack problem [1]. For example, a fully polynomial time approximation scheme for the 0-1 knapsack problem has been presented in [14]. Nonetheless, very few rigorous investigations addressing the approximation ratio of EAs on the 0-1 knapsack problem exist. [15] recast the 0-1 knapsack problem into a bi-objective knapsack problem with two conflicting objectives (maximizing profits and minimizing weights). A $(1+\epsilon)$ -approximate set of the knapsack problem has been introduced for the bi-objective optimization problem. An MOEA, called Restricted Evolutionary Multiobjective Optimizer, has been designed to obtain the $(1+\epsilon)$ -approximate set. A pioneering contribution of [15] is a rigorous runtime analysis of the proposed MOEA.

The current paper focuses on investigating the approximation ratio of three types of $(N+1)$ EAs combining bitwise mutation, truncation section and diverse repair mechanisms for the 0-1 knapsack problem. The first type is several pure strategy EAs, where a single repair method is exploited in the EAs. The second type is several mixed strategy EAs, which choose a repair method from a repair method pool randomly. The third type is a multi-objective EA using helper objectives, which is a simplified version of the EA in [16].

The remainder of the paper is organized as follows. The 0-1 knapsack problem is introduced in section II. In section III we analyse pure strategy EAs, while in section IV we analyse mixed strategy EAs. Section V is devoted to analysing an MOEA using helper objectives. Section VI concludes the article.

II. KNAPSACK PROBLEM AND APPROXIMATION SOLUTION

The 0-1 knapsack problem is the most important knapsack problem and one of the most intensively studied combinatorial optimisation problems [1]. Given an instance of the 0-1 knapsack problem with a set of weights w_i , profits p_i , and capacity C of a knapsack, the task is to find a binary vector $\vec{x} = (x_1 \cdots x_n)$ so as to

$$\begin{aligned} & \max_{\vec{x}} \sum_{i=1}^n p_i x_i, \\ & \text{subject to } \sum_{i=1}^n w_i x_i \leq C, \end{aligned} \quad (1)$$

where $x_i = 1$ if the item i is selected in the knapsack and $x_i = 0$ if the item i is not selected in the knapsack. A feasible solution is a knapsack represented by a binary vector $\vec{x} = (x_1 x_2 \cdots x_n)$ which satisfies the constraint. An infeasible one is an \vec{x} that violates the constraint. The vector $(0 \cdots 0)$ represents a null knapsack.

In last two decades, evolutionary algorithms, especially genetic algorithms (GAs), have been well adopted for tackling the knapsack problem [2], [3]. In order to assess the quality of solutions in EAs, we follow the classical α -approximation algorithm (see [13] for a detailed exposition) and define an evolutionary approximation algorithm as follows.

Definition 1: We say that an EA is an α -approximation algorithm for an optimization problem if for all instances of the problem, the EA can produce a solution within a polynomial runtime, the value of which is within a factor of α of the value of an optimal solution, regardless of the initialization. Here the runtime is measured by the expected number of function evaluations.

For instance, in case of the 0-1 knapsack problem, an evolutionary $1/2$ -approximation algorithm always can find a solution the value of which is at least a half of the optimal value within a polynomial runtime.

III. PURE STRATEGY (N+1) EVOLUTIONARY ALGORITHMS

In this section we analyze pure strategy $(N+1)$ EAs for the 0-1 knapsack problem. Here a *pure strategy EA* refers to an EA that employs a single repair method. The genetic operators used in $(N+1)$ EAs are bitwise mutation and truncation selection.

- *Bitwise Mutation:* Flip each bit with probability $1/n$.
- *Truncation Selection:* Select the best N individuals from the parent population and the child.

A number of diverse methods are available to handle constraints in EAs [6], [17]. Empirical results indicate that repair methods are more efficient than penalty function methods for the knapsack problem [18]. Thus, only the repair methods are investigated in the current paper. The repair procedure [6] is explained as follows.

```

1: input  $\vec{x}$ ;
2: if  $\sum_{i=1}^n x_i w_i > C$  then
3:    $\vec{x}$  is infeasible;
4:   while ( $\vec{x}$  is infeasible) do
5:      $i =$ : select an item from the knapsack;
6:     set  $x_i = 0$ ;
7:     if  $\sum_{i=1}^n x_i w_i \leq C$  then
8:        $\vec{x}$  is feasible;
9:     end if
10:  end while
11: end if
12: output  $\vec{x}$ .

```

There are several **select** methods available for the **repair** procedure, such as the *profit-greedy repair*, the *ratio-greedy repair* and the *random repair methods*.

- 1) *Profit-greedy repair:* sort the items x_i according to the decreasing order of their corresponding profits p_i . Then select the item with the smallest profit and remove it from the knapsack.
- 2) *Ratio-greedy repair:* sort the items x_i according to the decreasing order of the corresponding ratios p_i/w_i . Then select the item with the smallest ratio and remove it from the knapsack.
- 3) *Random repair:* select an item x_i from the knapsack at random and remove it from the knapsack.

Thanks to the repair method, all of the infeasible solutions have been repaired into the feasible ones. The fitness function of a feasible solution \vec{x} is $f(\vec{x})$.

First, let's consider a pure strategy $(N+1)$ EA using ratio-greedy repair for solving the 0-1 knapsack problem, which is described as follows.

- 1: **input** an instance of the 0-1 knapsack problem;
- 2: initialize a population considering of N individuals;
- 3: **for** $t = 0, 1, 2, \dots$ **do**
- 4: mutate one individual and generate a child;
- 5: **if** the child is an infeasible solution **then**

6: repair it into a feasible solution using the ratio-greedy repair;
7: **end if**
8: select N individuals from the parent population and the child using truncation selection;
9: **end for**
10: **output** the maximum of the fitness function.

The following proposition reveals that the $(N + 1)$ EA using the ratio-greedy repair cannot produce a good solution to the 0-1 knapsack problem within a polynomial runtime.

Proposition 1: For any constant $\alpha \in (0, 1)$, the $(N + 1)$ EA using Ratio-Greedy Repair is not an α -approximation algorithm for the 0-1 knapsack problem.

Proof: According to definition 1, it suffices to consider the following instance of the 0-1 knapsack problem:

Item i	1	$2, \dots, \alpha n$	$\alpha n + 1, \dots, n$
Profit p_i	n	$\frac{1}{\alpha n}$	$\frac{1}{n}$
Weight w_i	n	$\frac{1}{\alpha n}$	$\frac{1}{n}$
Capacity	n		

where without loss of generality, suppose αn is a large positive integer for a sufficiently large n .

The global optimum for the instance described above is

$$(10 \dots 0), \quad f(10 \dots 0) = n.$$

A local optimum is

$$(\overbrace{01 \dots 10}^{\alpha n - 1} \dots 0), \quad f(\overbrace{01 \dots 10}^{\alpha n - 1} \dots 0) = \alpha n - 1.$$

The ratio of fitness between the local optimum and the global optimum is

$$\frac{\alpha n - 1}{n} < \alpha.$$

Suppose that the $(N + 1)$ EA starts at the above local optimum having the 2nd highest fitness. Truncation selection combined with the ratio-greedy repair prevents a mutant solution from entering into the next generation unless the mutant individual is the global optimum itself. Thus, it arrives at the global optimum only if $\alpha n - 1$ one-valued bits are flipped into zero-valued ones and the bit x_1 is flipped from $x_i = 0$ to $x_i = 1$; other zero-valued bits remain unchanged. The probability of this event happening is

$$\left(\frac{1}{n}\right)^{\alpha n} \left(1 - \frac{1}{n}\right)^{n - \alpha n}.$$

Thus, we now deduce that the expected runtime is $\Omega(n^{\alpha n})$, that is exponential in n . This completes the argument. ■

Let the constant α towards 0, proposition 1 tells us that the solution produced by the $(N + 1)$ EA using the ratio-greedy repair after a polynomial runtime may be arbitrarily bad.

Next, we consider another pure strategy $(N + 1)$ EA that uses the random-greedy repair to tackle the 0-1 knapsack problem, which is described as follows.

1: **input** an instance of the 0-1 knapsack problem;
2: initialize a population considering of N individuals;
3: **for** $t = 0, 1, 2, \dots$ **do**
4: mutate one individual and generate a child;
5: **if** the child is an infeasible solution **then**
6: repair it into a feasible solution using the random-greedy repair;
7: **end if**
8: select N individuals from the parent population and the child using truncation selection;
9: **end for**
10: **output** the maximum of the fitness function.

Similarly, we may prove that this EA cannot produce a good solution to the 0-1 knapsack problem within a polynomial runtime using the same instance as that in Proposition 1.

Proposition 2: For any constant $\alpha \in (0, 1)$, the $(N + 1)$ EA using Random Repair is not an α -approximation algorithm for the 0-1 knapsack problem.

Proposition 2 tells us that the solution produced by the $(N + 1)$ EA using random repair is arbitrary bad.

Finally we investigate a pure strategy $(N + 1)$ EA using profit-greedy repair for solving the 0-1 knapsack problem, which is described as follows.

- 1: **input** an instance of the 0-1 knapsack problem;
- 2: initialize a population considering of N individuals;
- 3: **for** $t = 0, 1, 2, \dots$ **do**
- 4: mutate one individual and generate a child;
- 5: **if** the child is an infeasible solution **then**
- 6: repair it into a feasible solution using the profit-greedy repair;
- 7: **end if**
- 8: select N individuals from the parent population and the child using truncation selection;
- 9: **end for**
- 10: **output** the maximum of the fitness function.

Proposition 3: For any constant $\alpha \in (0, 1)$, the $(N + 1)$ EA using profit-greedy repair is not an α -approximation algorithm for the 0-1 knapsack problem.

Proof: Let's consider the following instance:

Item i	1	$2, \dots, n$
Profit p_i	$\alpha(n - 1)$	1
Weight w_i	$n - 1$	1
Capacity	n	

where without loss of generality, suppose $\alpha(n - 1)$ is a large positive integer for a sufficiently large n .

The local optimum is

$$(10 \dots 0), \quad f(10 \dots 0) = \alpha(n - 1).$$

and the global optimum is

$$(\overbrace{01 \dots 1}^n), \quad f(\overbrace{01 \dots 1}^n) = n - 1.$$

The fitness ratio between the local optimum and the global optimum is

$$\frac{\alpha(n - 1) - 1}{n - 1} < \alpha.$$

Suppose that the $(N + 1)$ EA starts at the local optimum $(10 \dots 0)$. Let's investigate the following mutually exclusive and exhaustive events:

- 1) An infeasible solution has been generated. In this case the infeasible solution will be repaired back to $(10 \dots 0)$ by profit-greedy repair.
- 2) A feasible solution having the fitness smaller than $\alpha(n - 1)$ has been generated. In this case, truncation selection will prevent the new feasible solution from being accepted.
- 3) A feasible solution is generated having fitness not smaller than $\alpha(n - 1)$. This is the only way in which truncation selection will preserve the new mutant solution. Nonetheless, this event happens only if the first bit of the individual in the initial population, Φ_0 , is flipped from $x_1 = 1$ into $x_1 = 0$ while at least $\alpha(n - 1)$ zero-valued bits of this individual, are flipped from $x_i = 0$ into $x_i = 1$. The probability of this event is

$$\begin{aligned} & \sum_{k=\alpha(n-1)}^n \frac{1}{n} \binom{n}{k} \left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{n-1-k} \\ & \leq O\left(\frac{e}{\alpha(n-1)}\right)^{\alpha(n-1)}. \end{aligned}$$

It follows immediately that if the EA starts at the local optimum $(10 \dots 0)$, the expected runtime to produce a better solution is exponential in n . The desired conclusion now follows immediately from definition 1. \blacksquare

Proposition 3 tells us that a solution produced by the $(N + 1)$ EA using profit-greedy repair may be arbitrarily bad as well.

In summary, we have demonstrated that none of the three pure strategy $(N + 1)$ EAs is an α -approximation algorithm for the 0-1 knapsack problem given any constant $\alpha \in (0, 1)$.

IV. MIXED STRATEGY (N+1) EVOLUTIONARY ALGORITHM

In this section we analyse mixed strategy evolutionary algorithm which combines several repair methods together. Here a *mixed strategy EA* refers to an EA employing two or more repairing methods selected with respect to a probability distribution over the set of repairing methods. It may be worth noting that other types of mixed strategy EAs have been considered in the

literature. For example, the mixed strategy EA in [19] employs four mutation operators. Naturally, we want to know whether or not a mixed strategy (1+1) EA, combining two or more repair methods together, may produce an approximation solution with a guarantee to the 0-1 knapsack problem.

A mixed strategy (1+1) EA for solving the 0-1 knapsack problem is described as follows. The EA combines both, ratio-greedy and profit-greedy repair methods together.

- 1: **input** an instance of the 0-1 knapsack problem;
- 2: initialize a population considering of N individuals;
- 3: **for** $t = 0, 1, 2, \dots$ **do**
- 4: mutate one individual and generate a child;
- 5: **if** the child is an infeasible solution **then**
- 6: select either ratio-greedy repair or profit-greedy repair method uniformly at random;
- 7: repair it into a feasible solution;
- 8: **end if**
- 9: select N individuals from the parent population and the child using truncation selection;
- 10: **end for**
- 11: **output** the maximum of the fitness function.

Unfortunately the quality of solutions in the mixed strategy EA still has no guarantee.

Proposition 4: Given any constant $\alpha \in (0, 1)$, the mixed strategy $(N + 1)$ EA using ratio-greedy repair and profit-greedy repair is not an α -approximation algorithm for the 0-1 knapsack problem.

Proof: consider the same instance as that in the proof of Proposition 3:

Item i	1	$2, \dots, n$
Profit p_i	$\alpha(n - 1)$	1
Weight w_i	$n - 1$	1
Capacity	n	

where the local optimum is $(10 \dots 0)$ and $f(10 \dots 0) = \alpha(n - 1)$. The global optimum is $(01 \dots 1)$ and $f(01 \dots 1) = n - 1$.

The fitness ratio between the local optimum and the global optimum is

$$\frac{\alpha(n - 1) - 1}{n - 1} < \alpha.$$

Suppose the $(N + 1)$ EA starts at the local optimum $(10 \dots 0)$. Let's analyse the following mutually exclusive and exhaustive events that occur upon completion of mutation:

- 1) A feasible solution is generated the fitness of which is smaller than $\alpha(n - 1)$. In this case, truncation selection will prevent the new feasible solution from entering the next generation.
- 2) A feasible solution is generated the fitness of which is not smaller than $\alpha(n - 1)$. The truncation selection may allow the new feasible solution to enter the next generation. This event happens only if the first bit is flipped from $x_1 = 1$ to $x_1 = 0$ and at least $\alpha(n - 1)$ zero-valued bits are flipped into one-valued. The probability of the event is then is

$$O\left(\frac{e}{\alpha(n - 1)}\right)^{\alpha(n - 1)}.$$

- 3) An infeasible solution is generated, but fewer than $\alpha(n - 1)$ zero-valued bits are flipped into the one-valued bits. In this case, either the infeasible solution will be repaired into $(10 \dots 0)$ through the profit-greedy repair; or, it is repaired into a feasible solution where $x_0 = 0$ and fewer than $\alpha(n - 1)$ one-valued bits among the rest of the bits through the ratio-greedy repair. In the later case the fitness of the new feasible solution is smaller than $\alpha(n - 1)$ and, therefore, cannot be accepted by the truncation selection.
- 4) An infeasible solution is generated but no fewer than $\alpha(n - 1)$ zero-valued bits are flipped into the one-valued bits. This event happens only if at least $\alpha(n - 1)$ zero-valued bits are flipped into the one-valued bits. The probability of the event is then is

$$O\left(\frac{e}{\alpha(n - 1)}\right)^{\alpha(n - 1)}.$$

Afterwards, with a positive probability, it is repaired into a feasible solution where $x_0 = 0$ and fewer than $\alpha(n - 1)$ one-valued bits among the rest of the bits by the ratio-greedy repair. In the later case the fitness of the new feasible solution is smaller than $\alpha(n - 1)$ and, therefore, it is prevented from entering the next generation by the truncation selection.

Summarizing the four cases described above, we see that when the EA starts at the local optimum $(10 \cdots 0)$, it is possible to generate a better solution with probability is

$$O\left(\frac{e}{\alpha(n-1)}\right)^{\alpha(n-1)}.$$

We then know that the expected runtime to produce a better solution is exponential in n . The conclusion of proposition 4 now follows at once. ■

Proposition 4 above tells us that solutions produced by the mixed strategy $(M+1)$ EA exploiting the ratio-greedy repair and profit-greedy repair may be arbitrarily bad.

Furthermore, we can prove, that even the mixed strategy $(N+1)$ EA combining the ratio-greedy repair, profit-greedy repair and random-repair together, is not an α -approximation algorithm for the 0-1 knapsack problem. Its proof is practically identical to that of Proposition 4.

In summary, we have demonstrated that mixed strategy $(N+1)$ EAs are α -approximation algorithms for the 0-1 knapsack problem given any constant $\alpha \in (0, 1)$.

V. MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM

So far, we have established several negative results about $(N+1)$ EAs for the 0-1 knapsack problem. A naturally arising important question is then how we can construct an evolutionary approximation algorithm. The most straightforward approach is to apply an approximation algorithm first to produce a good solution, and, afterwards, to run an EA to seek the global optimum solution. Nonetheless, such EAs sometimes get trapped into the absorbing area of a local optimum, so it is less efficient in seeking the global optimum.

Here we analyse a multi-objective EA using helper objectives (denoted by MOEA in short), which is similar to the EA presented in [16], but small changes are made in helper objectives for the sake of analysis. Experiment results in [16] have shown that the MOEA using helper objectives performs better than the simple combination of an approximation algorithm and a GA.

The MOEA is designed using the *multi-objectivization* technique. In multi-objectivization, single-objective optimisation problems are transferred into multi-objective optimisation problems by decomposing the original objective into several components [20] or by adding helper objectives [21]. Multi-objectivization may bring both positive and negative effects [22]–[24]. This approach has been used for solving several combinatorial optimisation problems, for example, the knapsack problem [15], vertex cover problem [25] and minimum label spanning tree problem [26].

Now we describe the MOEA using helper objectives, similar to the EA in [16]. The original single objective optimization problem (1) is recast into a multi-objective optimization problem using three helper objectives. First let's look at the following instance.

Item	1	2	3	4	5
Profit	10	10	10	12	12
Weight	10	10	10	10	10
Capacity	20				

The global optimum is 00011 in this instance. In the optimal solution, the average profit of packed items is the largest. Thus the first helper objective is to maximize the average profit of items in a knapsack. We don't use the original value of profits, instead we use the ranking value of profits. Assume that the profit of item i is the k th smallest, then let the ranking value $\hat{p}_i = k$. For example in the above instance, $\hat{p}_1 = \hat{p}_2 = \hat{p}_3 = 1$ and $\hat{p}_4 = \hat{p}_5 = 2$. Then the helper objective function is defined to be

$$h_1(\vec{x}) = \frac{1}{\|\vec{x}\|_1} \sum_{i=1}^n x_i \hat{p}_i, \quad (2)$$

where $\|\vec{x}\|_1 = \sum_{i=1}^n x_i$.

Next we consider another instance.

Item	1	2	3	4	5
Profit	15	15	20	20	20
Weight	10	10	20	20	20
Capacity	20				

The global optimum is 11000 in this instance. In the optimal solution, the average profit-to-weight ratio of packed items is the largest. However, the average profit of these items is not the largest. Then the second helper objective is to maximize the average profit-to-weight ratio of items in a knapsack. We don't use the original value of profit-to-weight, instead its ranking

value. Assume that the profit-to-weight of item i is the k th smallest, then let the ranking value $\hat{r}_i = k$. For example in the above instance, $\hat{r}_1 = \hat{r}_2 = 2$ and $\hat{r}_3 = \hat{r}_4 = \hat{r}_5 = 1$. Then the helper objective function is defined to be

$$h_2(\vec{x}) = \frac{1}{\|\vec{x}\|_1} \sum_{i=1}^n x_i \hat{r}_i. \quad (3)$$

Finally let's see the following instance.

Item	1	2	3	4	5
Profit	40	40	40	40	150
Weight	30	30	30	30	100
Capacity	120				

The global optimum is 11110 in this instance. In the optimal solution, neither the average profit of packed items nor average profit-to-weight ratio is the largest. Instead the number of packed items is the largest, or the average weight is the smallest. Thus the third helper objectives are to maximize the number of items in a knapsack. The objective functions are

$$h_3(\vec{x}) = \|\vec{x}\|_1. \quad (4)$$

We then consider a multi-objective optimization problem:

$$\begin{aligned} & \max_{\vec{x}} \{f(\vec{x}), h_1(\vec{x}), h_2(\vec{x}), h_3(\vec{x})\}, \\ & \text{subject to } \sum_{i=1}^n w_i x_i \leq C. \end{aligned} \quad (5)$$

The multi-objective optimisation problem (5) is solved by an EA using bitwise mutation, and multi-criteria truncation selection, plus a mixed strategy of two repair methods.

- 1: **input** an instance of the 0-1 knapsack problem;
- 2: initialize a population considering of N individuals;
- 3: **for** $t = 0, 1, 2, \dots$ **do**
- 4: mutate one individual and generate a child;
- 5: **if** the child is an infeasible solution **then**
- 6: select either ratio-greedy repair or profit-greedy repair method uniformly at random;
- 7: repair it into a feasible solution;
- 8: **end if**
- 9: select N individuals from the parent population and the child using the multi-criterion truncation selection;
- 10: **end for**
- 11: **output** the maximum of the fitness function.

A novel *multi-criteria truncation selection operator* is adopted in the above EA. Since the target is to maximise several objectives simultaneously, we select a few individuals which have higher function values with respect to each objective function. The pseudo-code of multi-criteria selection is described as follows.

- 1: **input** the parent population and the child;
- 2: merge the parent population and the child into a temporary population which consists of $N + 1$ individuals;
- 3: sort all individuals in the temporary population in the descending order of $f(\vec{x})$, denote them by $\vec{x}_1^{(1)}, \dots, \vec{x}_{N+1}^{(1)}$;
- 4: select all individuals from left to right (denote them by $\vec{x}_{k_1}^{(1)}, \dots, \vec{x}_{k_m}^{(1)}$) which satisfy $h_1(\vec{x}_{k_i}^{(1)}) < h_1(\vec{x}_{k_{i+1}}^{(1)})$ or $h_2(\vec{x}_{k_i}^{(1)}) < h_2(\vec{x}_{k_{i+1}}^{(1)})$ for any k_i .
- 5: **if** the number of selected individuals is greater than $\frac{N}{3}$ **then**
- 6: truncate them to $\frac{N}{3}$ individuals;
- 7: **end if**
- 8: add the selected individuals into the next generation population;
- 9: resort all individuals in the temporary population in the descending order of $h_1(\vec{x})$, still denote them by $\vec{x}_1, \dots, \vec{x}_{N+1}$;
- 10: select all individuals from left to right (still denote them by $\vec{x}_{k_1}, \dots, \vec{x}_{k_m}$) which satisfy $h_3(\vec{x}_{k_i}) < h_3(\vec{x}_{k_{i+1}})$ for any k_i .
- 11: **if** the number of selected individuals is greater than $\frac{N}{3}$ **then**
- 12: truncate them to $\frac{N}{3}$ individuals;
- 13: **end if**
- 14: add the selected individuals into the next generation population;
- 15: resort all individuals in the temporary population in the descending order of $h_2(\vec{x})$, still denote them by $\vec{x}_1, \dots, \vec{x}_{N+1}$;
- 16: select all individuals from left to right (still denote them by $\vec{x}_{k_1}, \dots, \vec{x}_{k_m}$) which satisfy $h_3(\vec{x}_{k_i}) < h_3(\vec{x}_{k_{i+1}})$ for any k_i .
- 17: **if** the number of selected individuals is greater than $\frac{N}{3}$ **then**
- 18: truncate them to $\frac{N}{3}$ individuals;

- 19: **end if**
- 20: add these selected individuals into the next generation population;
- 21: **while** the next generation population size is less than N **do**
- 22: randomly choose an individual from the parent population and child, and add it into the next generation population;
- 23: **end while**
- 24: **output** a new population Φ_{t+1} .

In the above algorithm, Steps 3-4 are for selecting the individuals with higher values of $f(\vec{x})$. In order to preserve diversity, we choose these individuals which have different values of $h_1(\vec{x})$ or $h_2(\vec{x})$. Similarly Steps 9-10 are for selecting the individuals with a higher value of $h_1(\vec{x})$. We choose the individuals which have different values of $h_3(\vec{x})$ for maintaining diversity. Steps 15-16 are for selecting individuals with a higher value of $h_2(\vec{x})$. Again we choose these individuals which have different values of $h_3(\vec{x})$ for preserving diversity. We don't explicitly select individuals based on $h_3(\vec{x})$. Instead we implicitly do it during Steps 9-10, and Steps 15-16.

Using helper objectives and multi-criterion truncation selection brings a benefit of searching along several directions $f(\vec{x})$, $h_1(\vec{x})$, $h_2(\vec{x})$ and implicitly $h_3(\vec{x})$. Hence the MOEA may arrive at a local optimum quickly, but at the same time, does not get trapped into the absorbing area of a local optimum of $f(\vec{x})$. The experiment results [16] have demonstrate the MOEA using helper objectives outperform the simplified combination of an approximation algorithm and a GA.

The analysis is based on a fact which is derived from the analysis of the greedy algorithm for the 0-1 knapsack problem (see [1, Section 2.4])). Consider the following algorithm:

- 1: let \vec{a}^* be the feasible solutions with the largest profit item;
- 2: resort all the items via the ratio of their profits to their corresponding weights so that $\frac{p_1}{w_1} \geq \dots \geq \frac{p_n}{w_n}$;
- 3: greedily add the items in the above order to the knapsack as long as adding an item to the knapsack does not exceeding the capacity of the knapsack. Denote the solution by \vec{b}^* .

Then the fitness of \vec{a}^* or \vec{b}^* is not smaller than 1/2 of the fitness of the optimal solution.

Based on the above fact, we can prove the following result.

Theorem 1: If $N \geq 3n$, then the $(N+1)$ MOEA can produce a feasible solution, which is not worse than \vec{b}^* and \vec{a}^* , within $O(Nn^3)$ runtime.

Proof: (1) Without loss of generality, let the first item be the most profitable one. First, it suffices to prove that the EA can generate a feasible solution fitting the Holland schema $(1 * \dots *)$ (as usual, $*$ stands for the 'don't care' symbol that could be replaced either by a 1 or a 0) within a polynomial runtime.

Suppose that the value of h_1 of all the individuals in the population are smaller than that of \vec{a}^* , that is, they fit the Holland schema $(0 * \dots *)$. Let \vec{x} be the individual that is chosen for mutation. Through mutation, x_1 can be flipped from $x_1 = 0$ to $x_1 = 1$ with probability $1/n$. If the child is feasible, then we arrive at the desired individual (denote it by \vec{y}). If the child is infeasible, then, with probability $1/2$, the first item will be kept thanks to the profit-greedy repair and a feasible solution is generated (denote it by \vec{y}). We have now shown that the EA can generate a feasible solution that includes the most profitable item with probability at least $1/(2n)$.

Thus, the EA can generate a feasible solution fitting the Holland schema $(1 * \dots *)$ within the expected runtime is at most $2n$.

(2) Without loss of generality, let

$$\frac{p_1}{w_1} > \dots > \frac{p_m}{w_m} > \dots > \frac{p_n}{w_n}.$$

and let $\vec{b}^* = (\overbrace{1 \dots 1}^m 0 \dots 0)$. We now demonstrate that the EA can reach \vec{b}^* within a polynomial runtime via objectives h_2 and h_3 .

First we prove that the EA can reach $(10 \dots 0)$ within a polynomial runtime. We exploit drift analysis [27] as a tool to establish the result. For a binary vector $\vec{x} = (x_1 \dots x_n)$, define the distance function

$$d(\vec{x}) = h_2(10 \dots 0) - h_2(\vec{x}). \quad (6)$$

For a population $(\vec{x}_1, \dots, \vec{x}_N)$, its distance function is

$$\min\{d(\vec{x}_1), \dots, d(\vec{x}_N)\}$$

According to the definition of $h_2(\vec{x})$, the above distance function is upper-bounded by n .

Suppose that none of individuals in the current population is $(10 \dots 0)$. Let \vec{x} be the individual, the value of whose distance is the smallest in the current population. The individual belongs to one of the two cases below:

Case 1: \vec{x} fits the Holland schema $(1 * \dots *)$ where at least one $*$ bit takes the value of 1.

Case 2: \vec{x} fits the Holland schema $(0 * \dots *)$.

The individual will be chosen for mutation with probability $\frac{1}{N}$. Now we analyse the mutation event related to the above two cases.

Analysis of Case 1: one of 1-valued *-bits (but not the first bit) is flipped into 0-valued; other bits are not changed. This event will happen with a probability

$$\frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1} = \Omega(n^{-1}). \quad (7)$$

Let's establish how the value of h_2 increases during the mutation. Denote the 1-valued bits in \vec{x} by i_1, \dots, i_k . Then the objective h_2 's value is

$$\frac{\hat{p}_{i_1} + \dots + \hat{p}_{i_k}}{k}.$$

Without loss of generality, the i_k th bit is flipped into 0-valued. Then after mutation, the 1-valued bits in \vec{x} becomes i_1, \dots, i_{k-1} and the objective h_2 's value is

$$\frac{\hat{p}_{i_1} + \dots + \hat{p}_{i_{k-1}}}{k-1}.$$

Thus, the value of h_2 increases (or equivalently, the value of d decreases) by

$$\frac{\hat{p}_{i_1} + \dots + \hat{p}_{i_{k-1}}}{k-1} - \frac{\hat{p}_{i_1} + \dots + \hat{p}_{i_k}}{k} = \Omega(n^{-2}). \quad (8)$$

Thanks to the multi-criteria truncation selection, the value of h_2 always increases. So there is no negative drift. Therefore the drift in Case 1 is

$$\Omega(N^{-1}n^{-3}). \quad (9)$$

Analysis of Case 2: The first bit is flipped into 0-valued; other bits are not changed. The analysis then is identical to Case 1. The drift in Case 2 is $\Omega(N^{-1}n^{-3})$, the same as that in Case 1.

Recall that the distance function $d(\vec{x}) \leq n$. Applying the drift theorem [27, Theorem 1], we deduce that the expected runtime to reach $(10 \dots 0)$ is $O(Nn^3)$. Once $(10 \dots 0)$ is included in the population, it will be kept for ever according to the multi-criteria truncation selection.

Next we prove that the EA can reach \vec{b}^* within a polynomial runtime when starting from $(10 \dots 0)$. Suppose that the current population includes an individual $(10 \dots 0)$ but no individual $(110 \dots 0)$. The individual $(10 \dots 0)$ may be chosen for mutation with a probability $\frac{1}{N}$, then it can be mutated into $(110 \dots 0)$ with a probability $\Omega(n^{-1})$. The individual $(110 \dots 0)$ has the second largest value of h_2 , thus, according to the multi-criteria truncation selection, it will be kept in the next generation population. Hence the expected runtime for the EA to reach the individual $(110 \dots 0)$ is $O(Nn)$. Similarly we can prove that the EA will reach $(1110 \dots 0)$ within $O(Nn)$ runtime, then $(11110 \dots 0)$ within $O(Nn)$ runtime, and so on. The expected runtime for the EA to reach \vec{b}^* is $O(Nn^2)$.

Combining the above discussions together, we see that the expected runtime to produce a solution not worse than \vec{a}^* and \vec{b}^* is $O(Nn^3) + O(Nn^2)$. ■

If we change helper objective functions $h_1(\vec{x})$ and $h_2(\vec{x})$ to those used in [16],

$$h_1(\vec{x}) = \frac{1}{\|\vec{x}\|_1} \sum_{i=1}^n x_i p_i, \quad (10)$$

$$h_2(\vec{x}) = \frac{1}{\|\vec{x}\|_1} \sum_{i=1}^n x_i \frac{p_i}{w_i}, \quad (11)$$

then the above proof doesn't work, and we need a new proof for obtaining the same conclusion. Furthermore, it should be mentioned that none of the three objectives can be removed; otherwise the MOEA will not produce a solution with a guaranteed approximation ratio. But on the other side, the performance might be better if adding more objectives, for example,

$$\min h_4(\vec{x}) = \frac{1}{\|\vec{x}\|_1} \sum_{i=1}^n x_i w_i. \quad (12)$$

VI. CONCLUSIONS

In this work, we have assessed the solution quality in three types of $(N+1)$ EAs, which exploit bitwise mutation and truncation selection, for solving the knapsack problem. We have proven that the pure strategy EAs using a single repair method and the mixed strategy EA combining two repair methods are not a α -approximation algorithm for any constant $\alpha \in (0, 1)$. In other words, solution quality in these EAs may be arbitrarily bad. Nevertheless, we have shown that a multi-objective $(N+1)$ EA using helper objectives is a $1/2$ -approximation algorithm. Its runtime is $O(Nn^3)$. Our work demonstrates that using helper objectives is a good approach to design evolutionary approximation algorithms. The advantages of the EA using helper objectives is to search along several directions and also to preserve population diversity.

Population-based EAs using other strategies of preserving diversity, such as niching methods, are not investigated in this paper. The extension of this work to such EAs will be the future research. Another work in the future is to study the solution quality of MOEAs for the multi-dimension knapsack problem.

Acknowledgements: This work was supported by the EPSRC under Grant No. EP/I009809/1 and by the NSFC under Grant No. 61170081.

REFERENCES

- [1] S. Martello and P. Toth, *Knapsack Problems*. Chichester: John Wiley & Sons, 1990.
- [2] Z. Michalewicz and J. Arabas, "Genetic algorithms for the 0/1 knapsack problem," in *Methodologies for Intelligent Systems*. Springer, 1994, pp. 134–143.
- [3] S. Khuri, T. Bäck, and J. Heitkötter, "The zero/one multiple knapsack problem and genetic algorithms," in *Proceedings of the 1994 ACM Symposium on Applied Computing*. ACM, 1994, pp. 188–193.
- [4] P. C. Chu and J. E. Beasley, "A genetic algorithm for the multidimensional knapsack problem," *Journal of Heuristics*, vol. 4, no. 1, pp. 63–86, 1998.
- [5] G. R. Raidl, "An improved genetic algorithm for the multiconstrained 0-1 knapsack problem," in *Proceedings of the 1998 IEEE World Congress on Computational Intelligence*. IEEE, 1998, pp. 207–211.
- [6] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. New York: Springer Verlag, 1996.
- [7] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.
- [8] A. Jaszkiewicz, "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem—a comparative experiment," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 402–412, 2002.
- [9] M. Eugénia Captivo, J. Climaco, J. Figueira, E. Martins, and J. Luis Santos, "Solving bicriteria 0–1 knapsack problems using a labeling algorithm," *Computers & Operations Research*, vol. 30, no. 12, pp. 1865–1886, 2003.
- [10] E. Özcan and C. Başaran, "A case study of memetic algorithms for constraint optimization," *Soft Computing*, vol. 13, no. 8, pp. 871–882, 2009.
- [11] R. Kumar and P. K. Singh, "Assessing solution quality of biobjective 0-1 knapsack problem using evolutionary and heuristic algorithms," *Applied Soft Computing*, vol. 10, no. 3, pp. 711–718, 2010.
- [12] P. Rohlfshagen and J. A. Bullnaria, "Nature inspired genetic algorithms for hard packing problems," *Annals of Operations Research*, vol. 179, no. 1, pp. 393–419, 2010.
- [13] D. P. Williamson and D. B. Shmoys, *The Design of Approximation Algorithms*. Cambridge University Press, 2011.
- [14] O. H. Ibarra and C. E. Kim, "Fast approximation algorithms for the knapsack and sum of subset problems," *Journal of the ACM*, vol. 22, no. 4, pp. 463–468, 1975.
- [15] R. Kumar and N. Banerjee, "Analysis of a multiobjective evolutionary algorithm on the 0–1 knapsack problem," *Theoretical Computer Science*, vol. 358, no. 1, pp. 104–120, 2006.
- [16] J. He, F. He, and H. Dong, "A novel genetic algorithm using helper objectives for the 0-1 knapsack problem," *arXiv*, vol. 1404.0868, 2014.
- [17] C. Coello and A. Carlos, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11–12, pp. 1245–1287, 2002.
- [18] J. He and Y. Zhou, "A comparison of GAs using penalizing infeasible solutions and repairing infeasible solutions II," in *Proceedings of the 2nd International Symposium on Intelligence Computation and Applications*. Wuhan, China: Springer, 2007, pp. 102–110.
- [19] J. He, W. Hou, H. Dong, and F. He, "Mixed strategy may outperform pure strategy: An initial study," in *Proceedings of 2013 IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 562–569.
- [20] J. D. Knowles, R. A. Watson, and D. W. Corne, "Reducing local optima in single-objective problems by multi-objectivization," in *Evolutionary Multi-Criterion Optimization*. Springer, 2001, pp. 269–283.
- [21] M. T. Jensen, "Helper-objectives: Using multi-objective evolutionary algorithms for single-objective optimisation," *Journal of Mathematical Modelling and Algorithms*, vol. 3, no. 4, pp. 323–347, 2005.
- [22] J. Handl, S. C. Lovell, and J. Knowles, "Multiobjectivization by decomposition of scalar cost functions," in *Parallel Problem Solving from Nature—PPSN X*. Springer, 2008, pp. 31–40.
- [23] D. Brockhoff, T. Friedrich, N. Hebbinghaus, C. Klein, F. Neumann, and E. Zitzler, "On the effects of adding objectives to plateau functions," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 591–603, 2009.
- [24] D. F. Lochtefeld and F. W. Ciarallo, "Helper-objective optimization strategies for the job-shop scheduling problem," *Applied Soft Computing*, vol. 11, no. 6, pp. 4161–4174, 2011.
- [25] T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, and C. Witt, "Approximating covering problems by randomized search heuristics using multi-objective models," *Evolutionary Computation*, vol. 18, no. 4, pp. 617–633, 2010.
- [26] X. Lai, Y. Zhou, J. He, and J. Zhang, "Performance analysis of evolutionary algorithms for the minimum label spanning tree problem," *IEEE Transactions on Evolutionary Computation*, 2014, (accepted, online).
- [27] J. He and X. Yao, "Drift analysis and average time complexity of evolutionary algorithms," *Artificial Intelligence*, vol. 127, no. 1, pp. 57–85, 2001.